Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

A New Way to Build a Very Fast Binary Adder

Lucian Jurca[1], Valentin Maranescu[2]

Abstract – This article presents a new way to build a very fast binary adder and the simulation results for a 32-b adder which confirmed that the operation speed is superior to other types of adders found in field literature. The adder is organised in two hierarchical levels. The superior level presents a carry select adder structure whose component blocks are 8-b carry look-ahead adders, which form the inferior level. Adopting the carry select mechanism and the new manner in which that was implemented allowed the maximum exploitance of the specific features of a carry look-ahead adder.

Keywords: carry look-ahead adder, carry select adder, arithmetic processor

I. INTRODUCTION

Whatever the type of an arithmetic processor, i.e. the type of the processed data, finally any operation is reduced to a basic one with integer numbers. The more frequent basic operation is the addition and all the other operations are executed by means of that one. For example, the subtraction is performed by adding the two's complement of the subtrahend to the first term. In a floating-point coprocessor, the multiplication is performed by adding the partial products and the division includes repetitive subtractions interleaved by comparisons. In a logarithmic processor the multiplication and the division are performed directly by means of the addition and the subtraction. The calculation of the trigonometric and transcendental functions is made by using some quick convergent algorithms that need the 4 basic operations also.

This means that the binary adders with integer numbers are among the most frequently used hardware components in an arithmetic processor and that is why it is very important for them to work as fast as possible.

The n-bit carry propagate adder (CPA) is the simplest binary adder but the slowest as well. Such an adder contains n 1-b full adders and each cell provides a sum bit s_i and a carry bit c_{i+1} for the next cell in function of the ith bits of the two word which are summed up and of the carry generated by the previous cell.

$$s_i = a_i \oplus b_i \oplus c_i \qquad (1)$$

$$c_{i+1} = a_i b_i + b_i c_i + c_i a_i$$

Because the carry propagates from one cell to other passing through two logical levels, this means that 2n logical levels must be counted for the obtaining of the result. Using the following equations the reduction of this number of logical levels can be made:

$$c_{i+1} = g_i + p_i c_i, \text{ where } g_i = a_i b_i \text{ and } p_i = a_i + b_i \qquad (2)$$

Replacing step by step we obtain:

$$c_{i+1} = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \dots$$
$$\dots + p_i p_{i-1} \quad p_1 g_0 + p_i p_{i-1} \quad p_1 p_0 c_0 \qquad (3)$$

In this case for providing the result five logical levels are needed as follows: one logical level to form the p and g terms using equations (2), two logical levels to obtain the carry using equation (3) and other two for the sum using equation (1). The disadvantages of the structure which implements the equations (3) are that gates with very many inputs are necessary, resulting an irregular structure and too much wiring. This idea was used, however, to build the carry look-ahead adder (CLA), which has $\log_2 n$ logical levels, i.e. less than a carry propagate adder, and more than that of a simple regular structure [1]. Here, the g and p auxiliary signals are built step by step. We already have:

$$c_1 = g_0 + c_0 p_0$$

Similarly we obtain:

$$c_2 = G_{01} + P_{01} c_0$$

where $G_{01} = g_1 + p_1 g_0$ and $P_{01} = p_1 p_0$. G_{01} shows that the block consisting of the first two bits generates a carry and P_{01} shows that through this block a previously generated carry can be propagated. In general, considering i<j<k, we can write the following equations:

$$C_{k+1} = G_{ik} + P_{ik} c_i \qquad (4)$$
$$G_{ik} = G_{j+1,k} + P_{j+1,k} G_{ij} \qquad (5)$$
$$P_{ik} = P_{ij} \, P_{j+1,k} \qquad (6)$$

where $P_{ii} = p_i$ and $G_{ii} = g_i$.

The equation (4) can be interpreted as follows: a carry is generated at the output of the block consisting of the bits from the ith to the kth if the upper part of this block (including the bits of weight between j+1

[1,2] Facultatea de Electronică şi Telecomunicaţii, Bd. V. Pârvan Nr. 2, 300223 Timişoara, Departamentul Electronică Aplicată
e-mail: jlucian@etc.utt.ro, valentin.maranescu@etc.utt.ro

and k) generates a carry or if there is a carry generated by the lower part (including the bits of weight between i and j) which can be propagated through the upper part. Implementing the recursive equations (4), (5), (6), the classical variant of CLA was obtained, which is presented in fig. 1. Here the binary numbers which have to be added enter the upper part of the tree, flow downwards in order to be combined with c_0, and then the data flow upwards in order to have the bits of the sum calculated. The structures of blocks A and B are shown in fig. 2a. and 2b. Compared to a CPA, the CLA obtains a substantial increase in speed with a small investment in area. This is even more significant if we take into consideration that the number of bits of the operands is bigger.

Another type of fast binary adder is the carry select adder. This works according to the following principle: two CPA work in parallel in each of the blocks of the adder (the exception being the least significant block). One of the adders of each block has the input carry 0, and the other 1. When the carry input generated by the previous block is known the right sum is selected. In fig.3 is presented a 19-b carry select adder which contains four blocks. If a block needs k units of time to calculate k bits and one unit of time to provide the input carry for the next block, from 2 available output carry signals and one input carry of the previous block, this means that in order to

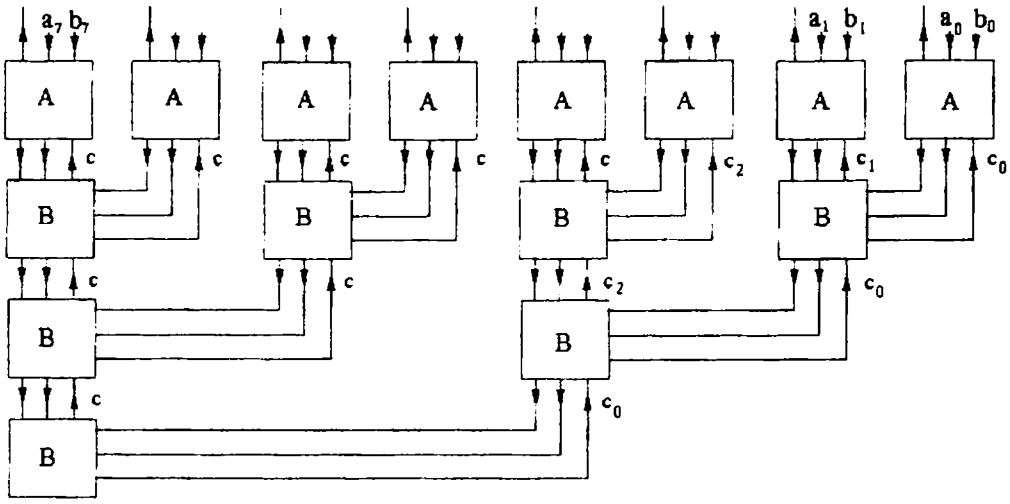

Fig. 1. Block diagram of an 8-b CLA

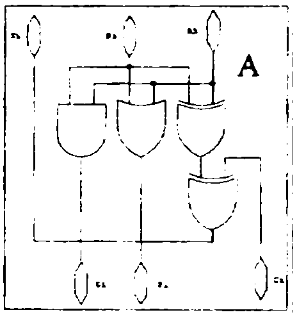

Fig 2a. Block A

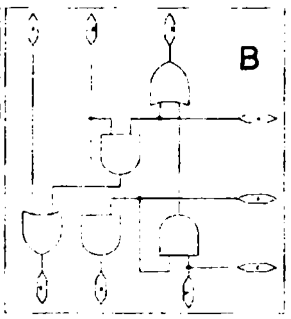

Fig. 2b. Block B

optimise the addition it is necessary for each block to be 1 bit larger than the previous one.

II. STRUCTURE OF THE NEW ADDER

If we analyse the structure of blocks A and B of the CLA, presented in fig. 2a and 2b, we can see that AND, OR and OR-exclusive gates are used. In CMOS technology the OR-exclusive gate has an optimised structure [2], while the other two types of gates are obtained by adding a CMOS inverter to the intrinsic gates NAND and NOR. This would means that for a

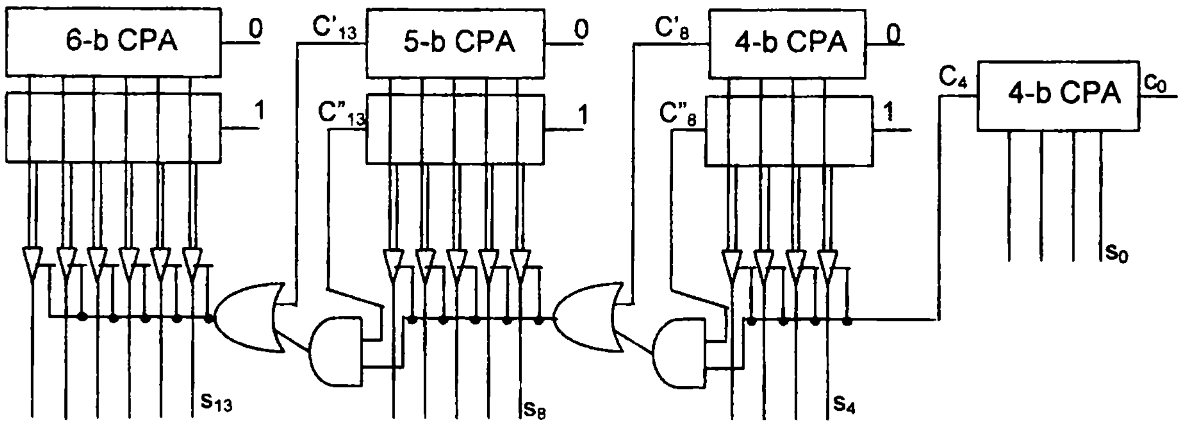

Fig. 3. A 19-b carry select adder

32-b CLA 219 inverters should be integrated, thus obviously increasing the area and the operation time.

As the field literature does not offer concrete solutions for the compensation of this disadvantage, in this chapter will be presented a new CLA with a modular structure of the type presented in fig. 1, which permits an optimal implementation in CMOS technology.

Thus, the structure of blocks A and B was modified as follows: the new block A' derived from block A will provide the p and g terms from the equations (2), practically complemented (fig. 4a). For restoring the logical level of the signals inside the blocks B, placed right under blocks A', CMOS inverters were added in a first stage. The further application of the De Morgan equations led to obtaining the B' blocks, whose structure is presented in fig. 4b. Because the logical state at the output of blocks B' is identical with that at blocks B, the second line of blocks type B from the previous diagram changes in the same manner as blocks A in order to provide complemented outputs, thus obtaining blocks of type B". In this way the logical compatibility with the third line of blocks, which will be type B', too, is insured. The fourth line of blocks will be of type B" and so on.

The block diagram of the 32-b adder maintains the same architecture but the structure of the component blocks and somewhat their arrangement will be different as shown in fig. 5.

This new CLA contains only NOR, NAND, OR-exclusive gates as well as CMOS inverters. Thus, the number of inverters was reduced from 219 to 62 and the gain in speed is more than 25%, as it will be shown in the chapter dealing with the simulation results.

The next step in this design was the analyses of the generation and propagation of data from the first 8-b block of the 32-b CLA in order to establish the input carry of the next 8-b block. A very important observation was made: the input carry of the second 8-b block (marked c_8 in fig. 5) is obtained significantly faster than some of the 8 bits of the first block. Indeed as it can be seen in fig. 5, the way to

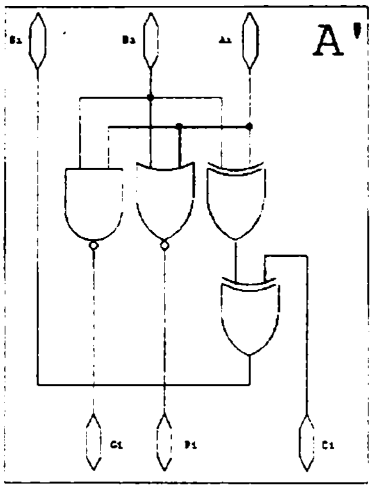

Fig. 4a. Block A'

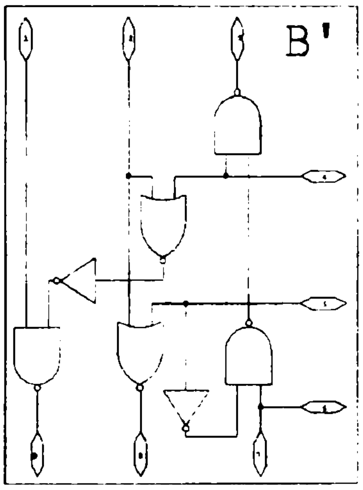

Fig. 4b. Block B'

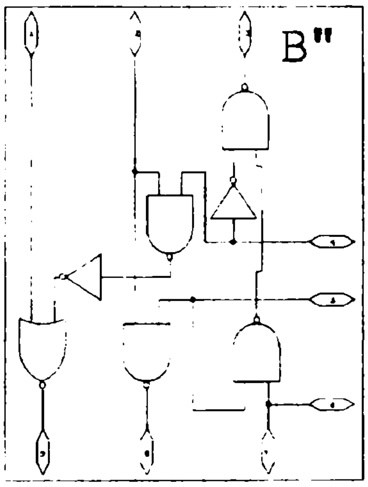

Fig. 4c. Block B"

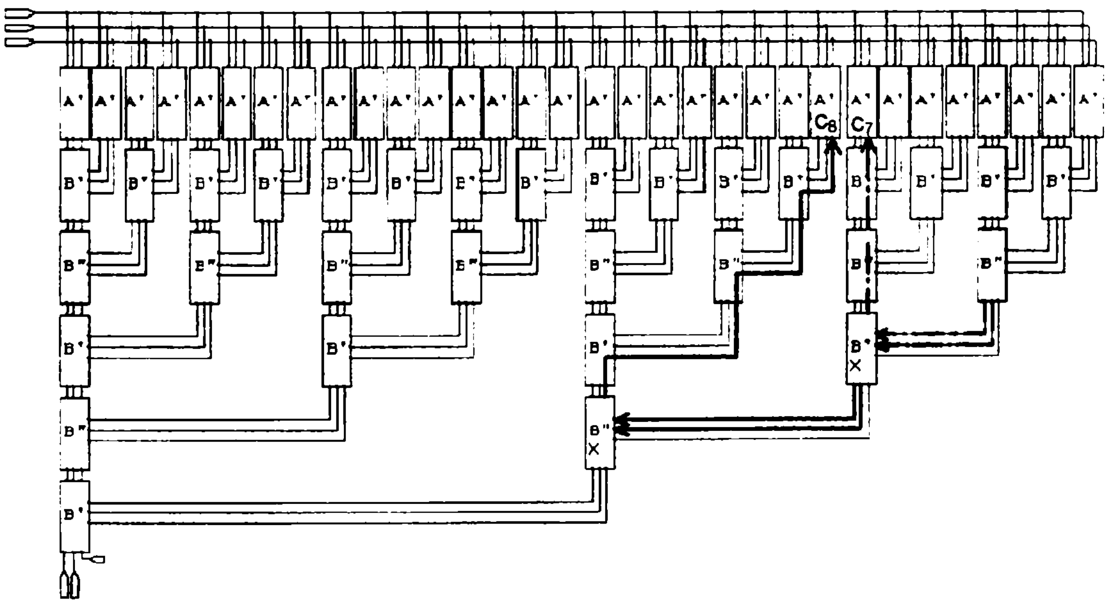

Fig. 5. Block diagram of the new 32-b CLA

generate c_8 from the output of block B'x, marked with a bolded continuous line, supposes the crossing of only two logical levels in block B'x. The other blocks on the way will be transparent on the formerly mentioned line, as it is shown in fig. 4b and 4c. On the other hand, the way for generating c_7, necessary for the calculation of bit s_7 of the sum, marked with a bolded dashed line, supposes the crossing of four logical levels through blocks B'' and B' from above B'x. To all this, other two logical levels are added in block A' for generating s_7 (see fig. 4a).

This property led to conceiving a new very fast adder organised in two hierarchical levels. The superior level presents a carry select adder structure whose component blocks are 8-b carry look-ahead adders, which form the inferior level. The block diagram of this adder is presented in fig. 6.

The blocks of type CONECTOR1 and 2 permit only a dissociation of the two 32-b input bus in four 8-b bus. The CONECTOR3 associates the four 8-b bus in only one bus of 32 bits.

The circuit contains a first CLA with 0 input carry and three pairs of CLA-s with 0 (blocks SUM8_Cin=0) respectively 1 (blocks SUM8_Cin=1) input carry. All the adders work simultaneously. When the c_8 carry from the output of the first adder is known, it will select through the SELECTOR block 1 the result from the adder of the first pair which had an input carry equal to c_8. In the same time it is also selected the right output carry, provided by the same adder. In its turn, this last signal will select through SELECTOR block 2 the right result and the right output carry from the second pair of adders. Because the c_8 type carry is obtained before the settling of all the output bits of the adders, the four 8-bit groups of the final sum are obtained almost simultaneously.

The SELECTOR blocks are made each with nine pairs of complementary pass transistors. In fact, each c_8 bit has to command in each SELECTOR block 18 gates of MOS transistors. The increasing of the control capability of selection lines with the help of CMOS inverter trees would lead to a significant delay. This drawback is compensated in great measure due to the above-emphasised property of the CLA. Furthermore, for the selection of the right input carry for the next pair of adders, a priority selection line was defined, the inverter tree beginning right behind it. This can be seen in fig. 7. Here the carry "C+in" selects "C+out" from "C+0" and "C+1" already produced, and after that the capability of the selection line is increased. Thus the speed of the 32-b adder will almost be that of an 8-b one.

Through the adopted selection mechanism, each CMOS output controls a maximum of 4 MOS gates. Because the inverter tree has an odd number of levels the selection logic of the multiplexer block MuxB is complemented in relation to that of the blocks Mux2-1.

III. SIMULATIONS RESULTS

In order to test the new circuits presented in chapter II the MicroSim program was used. In its device libraries the propagation delays through the different

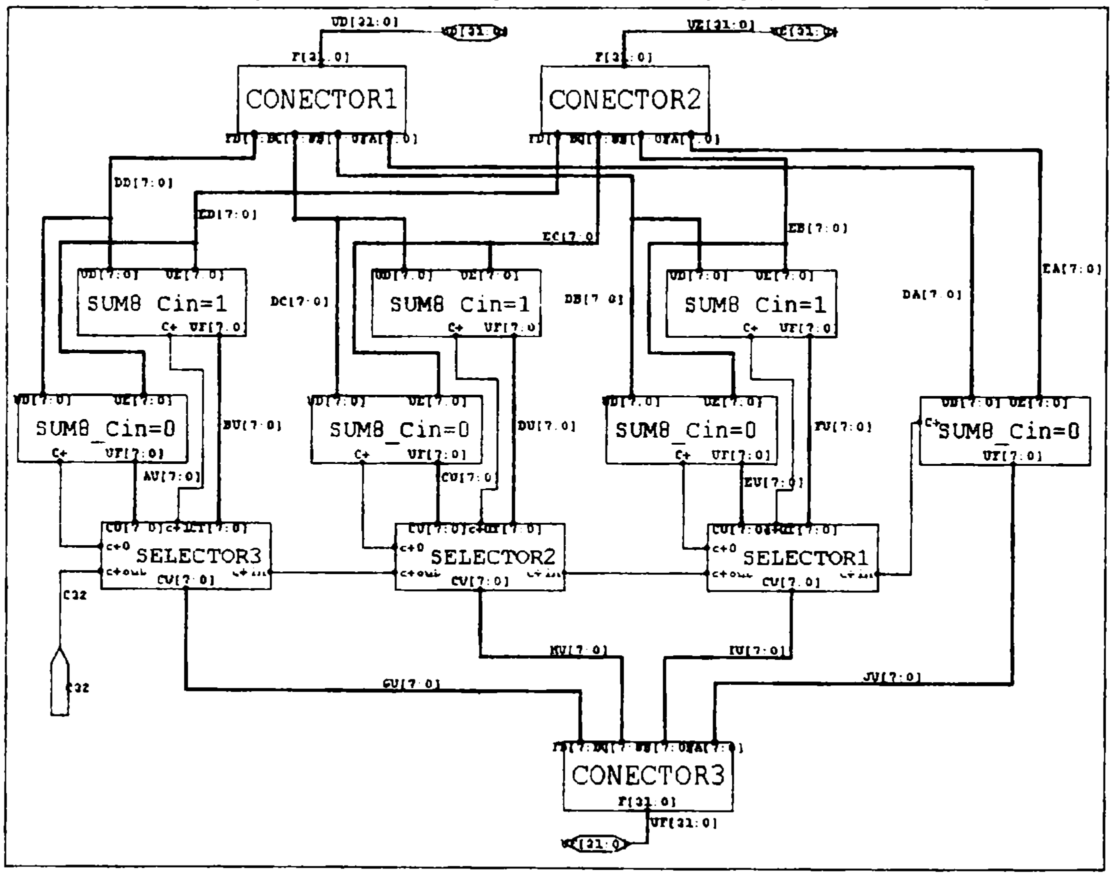

Fig. 6. Block diagram of the new very fast 32-b adder

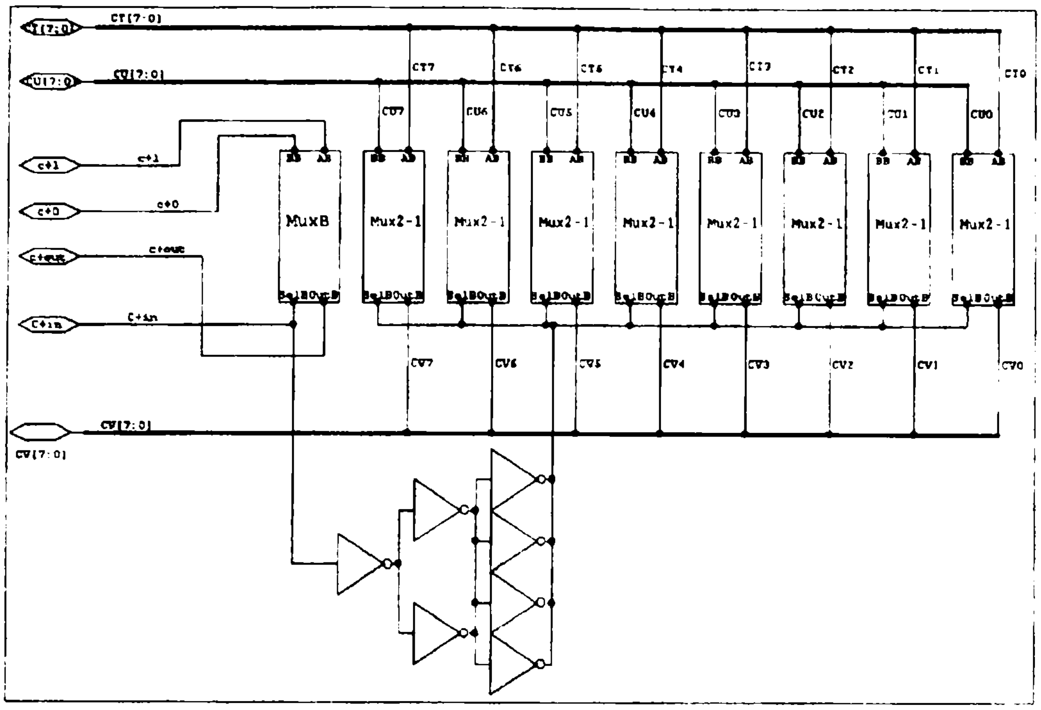

Fig. 7. Block diagram of the SELECTOR

logical gates were modified according to the information provided by [2], corresponding to a 0.5μm CMOS technology, adopted as reference to facilitate the comparison with similar circuits from the field literature. Thus, the propagation delays through the different logical gates were changed as follows: 0.35ns for unconventional OR-exclusive gates (faster than the classical ones), 0.2ns for 2-input NAND and NOR gates, 0.1ns for CMOS inverters and transmission gates.

For finding the most unfavourable case, i.e. the pair of operands for which the biggest delay is obtained, in the case of an 8-b CLA, a program written in C language was used. The program implements the recursive equations (4), (5), (6) and associates counters which are incremented at each change of the logical state after each logical level. After running the program, all the 65536 possible combinations of operands were tested and 256 pairs presenting the maximum result generation time were found. Among these pairs we found the one which corresponds to the most unfavourable case of the CPA, namely FFh and 01h. On the other hand, the case of FFh and 00h with input carry 1 does not represents one of the most unfavourable cases, a fact also confirmed by simulation.

The important conclusion that was drawn here was that, due to the modular structure of a CLA, we were able to extend this specific case as the most unfavourable case for larger CLA-s. For a 32-b CLA this means FFFFFFFFh and 00000001h. In the case of this adder, after the modifications, which were made, the maximum calculation time decreased from 5.7ns to 4.3ns (fig. 8). In the case of a 16-b CLA this propagation delay was 3.6ns, while for an 8-b CLA it was 2.5ns (fig. 9). In reality these times are 0.1ns smaller because the fact that the input carry of these adders is 0 in the most unfavourable case leads to a simplification of blocks B' and B" placed on the inferior frame of the modular structure of the CLA.

In terms of speed, the optimal structure of the new adder which implements the carry select mechanism was that in which the component blocks were 8-b CLA-s. This structure is optimal because the time period from the generation of the carry provided by the smallest weight CLA to the settling of the least

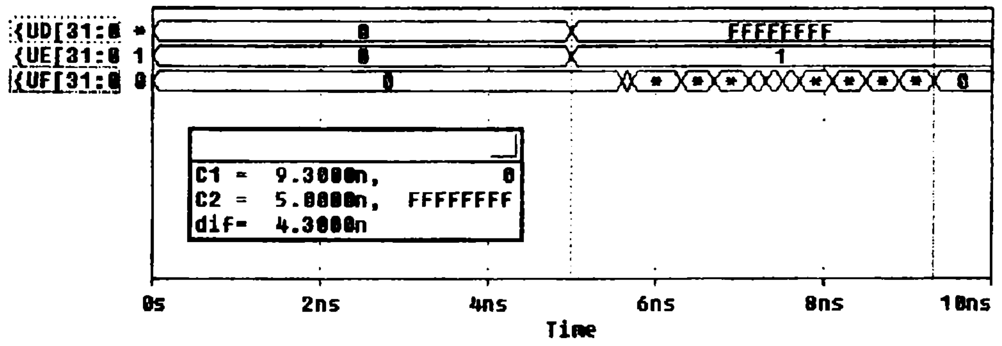

Fig. 8. Maximum propagation delay for the new 32-b CLA

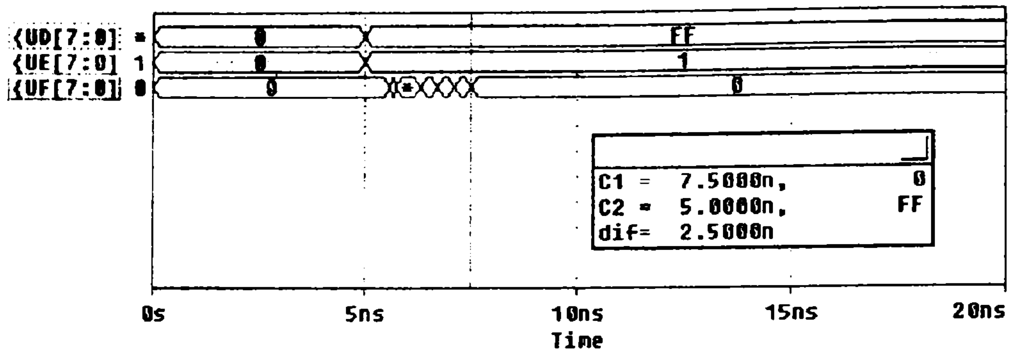

Fig. 9 Maximum propagation delay for the new 8-b CLA

significant 8 bits of the sum covers almost entirely the time necessary for the multiplication of the control capability of the selection lines of the right input carry for the bigger weight CLA-s. For this reason, the time necessary for producing the right result by this new 32-b adder is only 0.2ns bigger than that of an 8b-CLA. This can be noticed in fig. 10.

The combination of operands which leads to the most unfavourable case for this new adder, resulted after delimiting each of the two operands in four 8-b sections, was established on the basis of the following findings:

a. the biggest weight CLA (whose result settles the last) has to be in its most unfavourable case, which corresponds to feeding at its inputs the pair FFh and 01h;

b. the CLA-s which have input carry 0 can represent a critical path because feeding an input carry 1 can not be found in any of the combinations which represent the most unfavourable case for a CLA. On the other hand, these 8-b adders mustn't generate carry 1 in order to maintain the same unfavourable case at the next adder. Thus, the most unfavourable case corresponds to the pair FF7F7F7Fh and 01010101h. In compensation the selection lines for the right result from each pair of 8-b adders of the same weight always remain 0 and the selection is very fast. Therefor the specific structure of the new adder does not allow the component adders to be in the most unfavourable case at the same time;

c. the 8-b adders which have the input carry 1 can represent a critical path (although this does not represent a most unfavourable case for a CLA) because generating the output carry and thus the selection of the right result at the next pair of adders is delayed. Therefor in this case, too a benefic compensation of delays on the selection lines and 8-b adders takes place. This situation corresponds to the pair of operands FFFFFFFFh and 00000001h.

As shown in the diagram from fig. 10, both sets of operands, specified in paragraphs b. and c., lead to a maximum delay of 2.6ns. Any other combination of these sets leads to a faster calculation of the result.

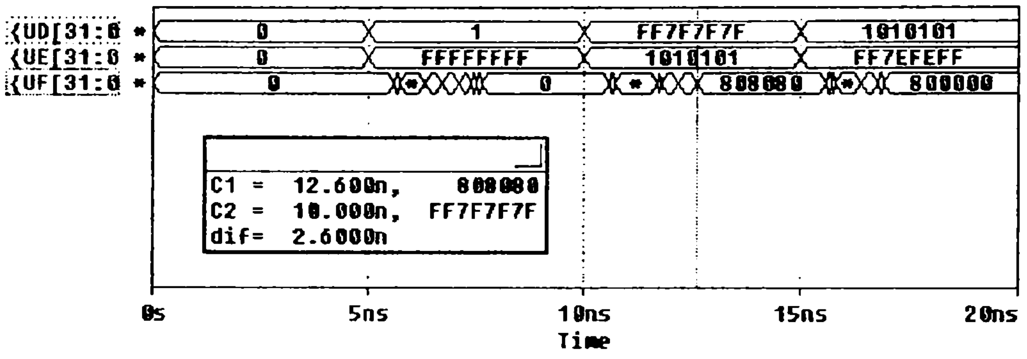

Fig. 10. Maximum propagation delay for the new very fast 32-b adder

Making a comparison in terms of speed with other adders presented in the field literature, synthesised in table 1, we can notice the superior speed of the proposed adder.

Table 1

Adder type	The new adder	[3]	[4]	[5]
Delay	2.6 ns	5ns	4.2ns	3ns

REFERENCES

[1] Stallings W., *Computer Architecture and Organization*, Prentice Hall Inc, 1996.

[2] Mori J., Nagamatsu M. *A 10-ns 54x54-b Parallel Structured Full Array Multiplier with 0.5-μm CMOS Tehnology*, IEEE Journal of Solid-State Circuits, Vol.26, No.4, April 1991.

[3] Lai F., *A 10-ns Hybrid Number System Data Execution Unit for Digital Signal Processing Systems*, IEEE Journal of Solid-State Circuits, Vol. 26, No. 4, Apr. 1991.

[4] Suzuki H. & all, *Leading-Zero Anticipatory Logic for High-Speed Floating Point Addition*, IEEE Journal of Solid-State Circuits, Vol. 31, No. 8, Aug. 1996.

[5] Senthinathan R. & all, *A 650-MHz IA-32 Microprocessor with Enhanced Data Streaming for Graphics and Video*, IEEE Journal of Solid-State Circuits, Vol. 34, No. 11, November 1999.